Horizons

S&P Global
Energy

451 Research Market
Insight Report Reprint

GenAI workload taxonomy

An early 2026 view

March 10, 2026

by Greg Macatee

While today's GenAI technology landscape remains highly dynamic, we identify and classify 42 individual workloads that best represent it. We also provide definitions of each of these workloads, as well as their parent workload categories, as a basis for analysis of various AI and machine learning systems, technologies and workflows.

Introduction

While today's GenAI technology landscape remains highly dynamic, we identify and classify 42 individual workloads that best represent it. We also provide definitions of each of these workloads, as well as their parent workload categories, as a basis for analysis of various AI and machine learning systems, technologies and workflows.

THE TAKE

The GenAI technology landscape remains a collection of "buzzword bingo" terms, with emerging tools, use cases and workflows seemingly impacting it on an everyday basis. Identifying and classifying its constituent workloads as part of a taxonomy research report allows us to analyze the current set of foundational AI workloads, provide a suggested framework to analyze current and future technologies, identify opportunities, and highlight potential challenges within existing applications and technology stacks. The GenAI landscape remains in constant flux, and we intend to update this report on an as-needed basis in alignment with ongoing technology trends and developments.

Context

According to 451 Research's Voice of the Enterprise: AI & Machine Learning, Infrastructure 2025 survey, organizations remain committed to deploying AI infrastructure and workloads in a variety of venues. We find that a plurality of them are following hybrid cloud strategies that represent a mixture of private and public cloud deployments, with penetration rates increasing as organizations grow and mature their GenAI capabilities.

GenAI workload deployment patterns follow similar trends when comparing model development and training stages with deployment and operationalization. For model training and development, 57% of organizations deploy infrastructure in on-premises data centers, 54% do so in managed service providers' data centers and 48% use public cloud. For model deployment and operationalization, 53% locate their infrastructure on-premises, 51% in MSP environments and 48% in the public cloud. These percentages decrease for infrastructure deployments in colocation facilities and at the edge (a noncore data center). Thirty-six percent (36%) of organizations use colocation facilities for model development and training, and 39% use these facilities for deployment and operationalization; 27% of them develop and train models at the edge, and 22% deploy and operationalize them there.

Market-level data from 451 Research's AI Infrastructure Market Monitor & Forecast provides further context about the AI workload opportunity. We project that AI infrastructure used to support data ingestion, integration and preparation workloads will expand from $97 billion in 2025 to $232 billion by 2029 (a four-year compound annual growth rate of 24%). Market opportunity for AI infrastructure used to support model training and fine-tuning workloads will increase from $111 billion to $223 billion (19% CAGR), while inferencing will grow from $91 billion to $395 billion (44% CAGR) over the same period.

GenAI workload taxonomy

We structure GenAI workloads into five main categories:

- **Data ingestion, integration and preparation** begin at the point where data is first collected and include its storage and transformation into useful formats to support a model throughout its life cycle.
- **Model training and fine-tuning** are the processes by which a model is trained from scratch or an existing model is augmented with new knowledge to teach specialized knowledge or specific behavior.

- **Model preparation and pre-inferencing** steps prepare a pre-trained or fine-tuned model for deployment with the goal of optimizing performance within its runtime environment. These steps also include optimization of infrastructure, specifically GPUs and other accelerators.
- **Inferencing and agentic AI** operationalize models, allowing them to make predictions or create generations based on new, previously unseen data and prompts. Agentic AI is an application of inferencing that allows software to act semi- or fully autonomously through a continuous loop of thought, action and observation.
- **Post-inferencing** occurs toward the end of the inferencing process and focuses on ensuring accuracy, safety and improvement of generations and other outputs.

These GenAI workload categories are composed of 42 individual GenAI workloads (Figure 1). We believe these underlying workloads are the fundamental building blocks of various AI systems and technologies, although they are subject to change over time.

Figure 1: GenAI workloads by category

GenAI workload categories	GenAI workloads
Data ingestion, integration and preparation	Ingestion and crawling Data integration Parsing and extraction Deduplication Quality filtering Data observability Sensitive data masking Schema mapping Metadata enrichment Data labeling, classification and annotation Feature engineering Chunking Embedding Vector indexing
Model training and fine-tuning	Pre-training Mid-training Continued pre-training Supervised fine-tuning Reinforcement learning Preference fine-tuning
Model preparation (pre-inference)	Compression Compilation Model evaluation Sharding and partitioning Model merging

GenAI workload categories	GenAI workloads
Inferencing and agentic AI	Tokenization Batch scheduling Embedding Hybrid search and retrieval Reranking Prefix cache lookup Key-value cache allocation Context prefill Real-time (general-purpose) inferencing Batch inferencing Reasoning Policy enforcement and guardrailing Orchestration Tool use and function calling State management
Post-inferencing	Aggregation Output parsing Hallucination detection Drift detection Feedback integration

Source: 451 Research from S&P Global Energy Horizons.

Data ingestion, integration and preparation

Data ingestion, integration and preparation workloads collect raw data from diverse sources — such as sensors, databases, APIs and the web — to store, transform, and enhance it for model training, fine-tuning and inferencing workloads.

Data ingestion focuses on the transport and storage of data, while integration, often executed via ETL or ELT pipelines, refines it into a usable format. Preparation includes cleaning (e.g., removing duplicates, handling missing values, masking personally identifiable information [PII]), normalizing (scaling) and feature engineering.

Ingestion and crawling involve connecting to external sources like the web, databases or APIs to pull data into a centralized storage repository, handling the initial acquisition of data, which can be streamed and processed continuously or grouped together in batches.

Data integration focuses on combining information from various disparate systems into a cohesive view. It typically involves generating patterns and workflows for ETL or ELT processes.

Parsing and extraction transform complex, non-machine-readable formats such as PDFs, HTML, images, audio and video into raw, machine-readable text, ensuring that underlying models can interpret and process ingested content.

Deduplication is the process of identifying and removing redundant data within a dataset. This prevents models from memorizing repeated information and improves overall processing efficiency.

Quality filtering cleans a dataset by stripping out inaccurate or meaningless information. It relies on lightweight heuristic models to automatically discard low-quality data before it is passed further down a pipeline.

Data observability provides continuous real-time monitoring of active data pipelines. It can act as an early warning system to detect anomalies, data outages and potential quality degradation as data flows through a system.

Sensitive data masking helps meet privacy requirements and can help better foster compliance by modifying datasets to obscure PII and other sensitive records.

Schema mapping bridges structural gaps between diverse data sources by detecting and aligning individual fields. It improves output accuracy and overall performance of the data pipeline by clearly identifying these relationships.

Metadata enrichment automatically extracts and attaches structured tags to unstructured data chunks. These metadata tags can help enable hybrid search capabilities, particularly in retrieval-augmented generation (RAG) systems.

Data labeling, classification, and annotation workloads involve tagging, classifying and categorizing raw data. These annotations provide necessary context to help models learn specific patterns and behaviors.

Feature engineering transforms raw data into more informative signals by creating new variables from existing data. This helps models better identify underlying relationships, which helps improve their accuracy.

Chunking breaks down longer inputs into smaller, semantically meaningful ones. This can help ensure inputs fit within a model's context window limits. It is a technique that is widely used when building RAG databases.

Embedding converts token IDs into numerical vectors that capture semantic meaning. It is an additional fundamental step when building RAG databases.

Vector indexing organizes semantic vector embeddings into a structured, searchable format for models and AI applications.

Model training and fine-tuning

Model training and fine-tuning workloads include processes for teaching and refining a machine learning (ML) model's ability to recognize patterns and relationships in training data, enabling it to make predictions (inferences).

For foundation models like large language models, training begins with pre-training, during which a model learns general language structure, facts and reasoning capabilities. During pre-training, a model starts with randomized weights (connections between neurons) that represent semantic meaning. These weights, along with their biases, are repeatedly optimized (minimizing mathematical error), which helps ensure consistency between a model's output and its training dataset.

Fine-tuning then further trains a pre-trained model on smaller, often domain-specific datasets. This specializes a model's knowledge or promotes specific behaviors and capabilities (e.g., reasoning) for targeted tasks.

Pre-training is a foundational phase of model development that encodes raw intelligence, general language capabilities, logic and broad knowledge into a model. During pre-training stages, a model continually adjusts its internal weights, biases, embeddings and attention mechanism to learn underlying patterns.

Mid-training is geared toward model optimization for a specific domain or task using curated, synthetic or known high-quality data. It occurs toward the end of a model's pre-training process, in which specialized skills or knowledge are injected before foundational training is finished.

Continued pre-training is a specialized training phase designed to inject deeper, domain-specific knowledge into an already established model. It takes place after pre-training and potential mid-training have been completed.

Supervised fine-tuning, or instruction tuning, teaches a model how to follow specific user instructions. It guides a model on how to appropriately structure, format and deliver its response to users or for other downstream tasks.

Reinforcement learning, or reasoning training, builds a model's internal logic and self-correction capabilities. This includes developing advanced behaviors like chain-of-thought reasoning, which allows these models to "think" and iterate through complex problems before producing a response.

Preference fine-tuning helps ensure a model's output is safe, high-quality and helpful. It specifically tunes a model's behavior to operate in a manner that aligns with human values and desired preferences.

Model preparation (pre-inference)

Model preparation and pre-inferencing workloads bridge the gap between a trained model and a production-ready application. Once model training and fine-tuning are complete, a model often exists as a large, raw file that can be computationally expensive and slow to run. Model preparation workloads focus on optimization and compilation to ensure that a model performs efficiently on specific hardware (e.g., GPUs, CPUs or edge devices). Compression, compilation and merging workloads operate at the model level and streamline overall performance by optimizing its structure or combining two or more models. Workloads like sharding and partitioning function at an infrastructure level, splitting computational and memory burdens across two or more accelerators.

Compression reduces the overall size of a model to improve performance by lowering computational and memory overhead. It relies on specific techniques like quantization, pruning and knowledge distillation to shrink a model's footprint while preserving its accuracy.

Compilation analyzes a model's internal computation graph and fuses its mathematical kernels for highly optimized execution. This process translates and tailors a model's operations for target hardware setup and configuration, ensuring maximum throughput.

Model evaluation utilizes automated adversarial attacks combined with standardized benchmarks to stress-test a system. It actively probes a model to uncover hidden safety flaws and verify accuracy before it is approved for operationalization. Model evaluation also may include red-teaming, which involves test users intentionally acting to try to break a model's safety guardrails.

Sharding and partitioning calculate how to split a model between two or more accelerators. By effectively distributing the computational load, they support the advanced parallelism necessary to run larger models that exceed a single chip's memory limits.

Model merging mathematically combines the weights of separate operational models into a single unified network. This allows developers to blend multiple specialized capabilities without the need for expensive and time-consuming retraining or fine-tuning.

Inferencing and agentic AI

Inferencing operationalizes trained models and deploys them to make predictions or generate output based on new, unseen data. Compared with model training and fine-tuning workloads, inferencing is significantly faster and less computationally intensive per transaction. It involves receiving an input (e.g., a prompt, query, image or sensor reading), processing it through a model's layers (i.e., forward passing), and outputting a probability score, classification or other generation. Low latency and high throughput are crucial metrics for inferencing, as real-time applications often necessitate immediate responses. Efficient inferencing relies heavily on the optimization steps taken during the model-preparation phase.

Agentic AI is an extension of inferencing and represents a paradigm shift from passive model outputs to autonomous systems pursuing complex goals. Unlike previous AI technologies, an agent perceives its environment, reasons through multi-step workflows and executes actions using external tools (e.g., APIs, web browsers and code interpreters). Agentic AI shifts from single-turn generation to orchestration, where a model acts as a central "brain" that directs other software to perform tasks.

Agentic AI operates through a continuous loop of thought, action and observation. An agent breaks down high-level objectives into smaller tasks, generates necessary code and commands to fulfill them, and evaluates results before proceeding. This requires sophisticated frameworks to manage memory (context retention over long and/or multiple sessions), planning (reasoning) and error handling, which allows an agent to self-correct if an action fails.

Tokenization converts human-perceptible inputs into a specific sequence of integer IDs that a model can process. It bridges the gap between raw text and numerical formats required for computation.

Batch scheduling optimizes hardware utilization by determining the most efficient way to fit multiple user requests into GPUs simultaneously. It helps maximize throughput by ensuring compute resources are not left sitting idle.

Embedding, as part of inferencing, converts token IDs into numerical vectors that capture semantic meaning. Unlike embeddings created during data ingestion and preparation stages, these vectors are passed directly through models (forward passing) to generate outputs.

Hybrid search and retrieval is used to search for relevant information by combining keyword matching with broader, concept-based semantic searches, and is frequently paired with a reranking step to maximize accuracy and relevancy of the retrieved context.

Reranking analyzes the relationships between a user's query and a set of potential search results. It identifies and prioritizes the exact information best suited to inform an output. Reranking is a technique commonly used in real-time inferencing and RAG alongside hybrid search and retrieval.

Prefix cache lookup checks an accelerator's memory to see if the needed context of a new request has already been processed in a previous turn. If a match is found (i.e., a cache hit occurs), it saves re-computation by reusing existing mathematical states.

Key-value cache allocation. KV cache allocation is a memory management workload that calculates and reserves specific blocks of memory (e.g., GPU VRAM) to store context. It acts as a model's short-term memory buffer.

Context prefill is a stage in which accelerators process prompts alongside other contextual data (e.g., from RAG) to understand specific context. This prepares a model's internal state before it begins generating a response.

Real-time (general-purpose) inferencing involves a model immediately processing a prompt and generating a response, one token at a time. It can take place in a variety of deployment locations, from on-premises data centers and the cloud to purpose-built edge deployments — increasingly including handheld devices. Real-time inferencing is often informed by RAG and other adjacencies like reranking.

Batch inferencing is an offline workload that processes datasets through a model all at once. The primary goal of batch inferencing is to maximize overall system throughput, hardware efficiency and costs rather than minimizing latency of a single response.

Reasoning allows a model to generate hidden chain-of-thought tokens before accepting and producing an output. By internally working through logic step by step, a system can self-correct and verify its own accuracy before outputting a result.

Policy enforcement and guardrailing are critical safety steps that involve running parallel classification models to continuously monitor and filter a model's behavior. They help ensure that all generated outputs are safe, appropriate, and aligned by blocking any unwanted or harmful responses.

Orchestration is a dynamic runtime process in which a model actively plans, executes and refines a sequence of actions to achieve a complex goal, particularly as part of agentic AI. It often acts as the central intelligence of a larger system, managing and delegating tasks to other specialized models or software entities.

Tool use and function calling allow a model to generate structured outputs like JSON so it can interact with other tools and functions via APIs. They are key components of agentic systems and enable a model to perform specific tasks (e.g., executing code or fetching live weather) that it cannot do on its own.

State management maintains the ongoing memory of an interaction by continuously tracking conversation history, context and intermediate variables across multiple turns. To retain application continuity without overloading a main model, it is often aided by separate inference models that actively compress and summarize the most relevant information.

Post-inferencing

Post-inferencing occurs immediately after a model generates an output but before it is presented to a user or downstream application, ensuring a model's response is safe, accurate and properly formatted. In advanced systems like RAG, post-inferencing often involves fact-checking and citation verification. This compares the generated response with the retrieved source documents to ensure the model did not hallucinate information.

Aggregation involves generating multiple potential responses for a prompt simultaneously. Once options are generated, a system algorithmically evaluates and selects the best token or response to ensure maximum quality.

Output parsing reformats a model's output so that other systems and applications can use it more easily. By converting output into predictable and structured formats, it ensures that downstream systems, APIs, and other applications can easily ingest and use the data without manual intervention.

Hallucination detection is a critical output verification process designed to catch and flag factually inaccurate or fabricated responses. It acts as a final quality assurance safeguard before a user or downstream application receives an output, helping to maintain trust and reliability in a system.

Drift detection provides continuous monitoring of a deployed model's health and relevance. It actively identifies when the live data a model processes diverges from its original training data (data drift) or when its overall accuracy begins to degrade over time (conceptual drift).

Feedback integration captures and processes user signals such as direct ratings, edits or behavioral interactions. These captured signals are then compiled into structured datasets that can be used for future fine-tuning and continuous model improvement.

CONTACTS

Americas: +1 800 597 1344
Asia-Pacific: +60 4 296 1125
Europe, Middle East, Africa: +44 (0) 203 367 0681

www.spglobal.com/energy
www.spglobal.com/en/enterprise/about/contact-us.html